In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
dataset=sns.load_dataset("iris")
dataset
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [3]:

```
dataset.iloc[50]
```

Out[3]:

```
sepal_length           7.0
sepal_width            3.2
petal_length           4.7
petal_width            1.4
species         versicolor
Name: 50, dtype: object
```

In [4]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:

```
dataset.isnull().sum()
```

Out[5]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [6]:

```
l1=LabelEncoder()
dataset["species"]=l1.fit_transform(dataset["species"])
```

In [7]:

```
dataset
# here 0 is for Setosa, 1 is for Versicolor 2 is for Virginica
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2

In [8]:

```
dataset["species"].value_counts()
# so data is balanced
```

Out[8]:

```
0    50
1    50
2    50
Name: species, dtype: int64
```

In [9]:

```
dataset.info()
# All information is now available in numeric
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [10]:

```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
```

In [11]:

```
print(x.shape)
```

```
(150, 4)
```

In [12]:

```
print(y.shape)
```

```
(150,)
```

In [13]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
print(Counter(y_test))
```

```
Counter({0: 14, 2: 8, 1: 8})
```

In [14]:

```
x_train = pd.DataFrame(x_train,columns=["sepal_length","sepal_width","petal_length","petal_
x_train
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width
0	6.2	2.8	4.8	1.8
1	5.1	3.3	1.7	0.5
2	5.6	2.9	3.6	1.3
3	7.7	3.8	6.7	2.2
4	5.4	3.0	4.5	1.5
...	...	...	...	...
115	6.6	3.0	4.4	1.4
116	5.0	3.5	1.6	0.6
117	4.6	3.6	1.0	0.2
118	6.3	2.5	4.9	1.5

In [15]:

```
print(x_train.shape)
```

```
(120, 4)
```

In [16]:

```
x_train.describe()
```

Out[16]:

	sepal_length	sepal_width	petal_length	petal_width
count	120.000000	120.00000	120.000000	120.000000
mean	5.897500	3.06000	3.861667	1.244167
std	0.813382	0.44616	1.742508	0.751894
min	4.300000	2.00000	1.000000	0.100000
25%	5.200000	2.80000	1.600000	0.400000
50%	5.800000	3.00000	4.450000	1.400000
75%	6.400000	3.40000	5.100000	1.825000
max	7.900000	4.40000	6.900000	2.500000

In [17]:

```
x_train.describe().round(2)
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width
count	120.00	120.00	120.00	120.00
mean	5.90	3.06	3.86	1.24
std	0.81	0.45	1.74	0.75
min	4.30	2.00	1.00	0.10
25%	5.20	2.80	1.60	0.40
50%	5.80	3.00	4.45	1.40
75%	6.40	3.40	5.10	1.82
max	7.90	4.40	6.90	2.50

In [18]:

```
x_test = pd.DataFrame(x_test,columns=["sepal_length","sepal_width","petal_length","petal_wi
x_test
```

13	5.4	3.9	1.7	0.4
14	6.0	3.4	4.5	1.6
15	6.5	2.8	4.6	1.5
16	4.5	2.3	1.3	0.3
17	5.7	2.9	4.2	1.3
18	6.7	3.3	5.7	2.5
19	5.5	2.5	4.0	1.3
20	6.7	3.0	5.0	1.7
21	6.4	2.9	4.3	1.3
22	6.4	3.2	5.3	2.3
23	5.6	2.7	4.2	1.3
24	6.3	2.3	4.4	1.3

In [19]:

```
x_test.describe()
```

Out[19]:

	sepal_length	sepal_width	petal_length	petal_width
count	30.000000	30.000000	30.000000	30.000000
mean	5.626667	3.046667	3.343333	1.020000
std	0.864604	0.398907	1.824674	0.789762
min	4.400000	2.300000	1.200000	0.100000
25%	4.800000	2.825000	1.500000	0.200000
50%	5.550000	3.000000	4.100000	1.300000
75%	6.400000	3.200000	4.975000	1.675000
max	7.200000	3.900000	6.000000	2.500000

In [20]:

```
print(x_test.shape)
```

```
(30, 4)
```

In [21]:

```
# sns.pairplot(x_train,)
```

In [22]:

```
logs=LogisticRegression()
logs.fit(x_train,y_train)
logs_pred=logs.predict(x_test)
cr = classification_report(y_test,logs_pred)
cm = confusion_matrix(y_test,logs_pred)
print("Report: ",cr)
print("Matrix: ",cm)
```

```
Report:                precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      0.88      0.93         8
           2       0.89      1.00      0.94         8

    accuracy                           0.97        30
   macro avg       0.96      0.96      0.96        30
weighted avg       0.97      0.97      0.97        30

Matrix:  [[14  0  0]
 [ 0  7  1]
 [ 0  0  8]]
```

Prediction with Scaled Data

In [23]:

```
scaler=StandardScaler()
x_train_sc=scaler.fit_transform(x_train)
x_test_sc=scaler.fit_transform(x_test)
```

In [24]:

```
x_train_sc = pd.DataFrame(x_train_sc,columns=["sepal_length","sepal_width","petal_length","
x_train_sc
```

Out[24]:

	sepal_length	sepal_width	petal_length	petal_width
0	0.373463	-0.585194	0.540754	0.742344
1	-0.984585	0.540179	-1.245751	-0.993873
2	-0.367290	-0.360119	-0.150796	0.074568
3	2.225348	1.665552	1.635708	1.276565
4	-0.614208	-0.135045	0.367866	0.341679
...	...	...	...	...
115	0.867299	-0.135045	0.310237	0.208124
116	-1.108044	0.990328	-1.303380	-0.860318
117	-1.601880	1.215403	-1.649155	-1.394539
118	0.496922	-1.260418	0.598383	0.341679
119	-0.243831	3.015999	-1.361009	-1.127429

120 rows × 4 columns

In [25]:

```
x_train_sc.describe()
```

Out[25]:

	sepal_length	sepal_width	petal_length	petal_width
count	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02
mean	1.163190e-15	1.872576e-15	4.033810e-16	5.995204e-16
std	1.004193e+00	1.004193e+00	1.004193e+00	1.004193e+00
min	-1.972257e+00	-2.385790e+00	-1.649155e+00	-1.528094e+00
25%	-8.611262e-01	-5.851939e-01	-1.303380e+00	-1.127429e+00
50%	-1.203725e-01	-1.350447e-01	3.390517e-01	2.081235e-01
75%	6.203812e-01	7.652535e-01	7.136413e-01	7.757331e-01
max	2.472265e+00	3.015999e+00	1.750966e+00	1.677231e+00

In [26]:

```
x_train_sc.describe().round(2)
# Scaling changes scale of data in such way that mean & variance of data becomes 0 & 1 resp
```

Out[26]:

	sepal_length	sepal_width	petal_length	petal_width
count	120.00	120.00	120.00	120.00
mean	0.00	0.00	0.00	0.00
std	1.00	1.00	1.00	1.00
min	-1.97	-2.39	-1.65	-1.53
25%	-0.86	-0.59	-1.30	-1.13
50%	-0.12	-0.14	0.34	0.21
75%	0.62	0.77	0.71	0.78
max	2.47	3.02	1.75	1.68

In [27]:

```
x_test_sc = pd.DataFrame(x_test_sc,columns=["sepal_length","sepal_width","petal_length","pe
x_test_sc
```

18	1.262638	0.645926	1.313634	1.906018
19	-0.149007	-1.393840	0.366034	0.360598
20	1.262638	-0.118986	0.923446	0.875738
21	0.909727	-0.373957	0.533257	0.360598
22	0.909727	0.390955	1.090669	1.648448
23	-0.031370	-0.883899	0.477516	0.360598
24	0.792090	-1.903782	0.588999	0.360598
25	-1.090104	0.390955	-0.971755	-1.056037
26	-1.090104	0.390955	-1.138978	-1.056037
27	0.556816	-0.118986	0.867705	1.004523
28	-0.619555	1.920780	-0.804531	-0.798467

In [28]:

```
x_test_sc.describe().round(2)
```

Out[28]:

	sepal_length	sepal_width	petal_length	petal_width
count	30.00	30.00	30.00	30.00
mean	0.00	-0.00	-0.00	-0.00
std	1.02	1.02	1.02	1.02
min	-1.44	-1.90	-1.19	-1.18
25%	-0.97	-0.57	-1.03	-1.06
50%	-0.09	-0.12	0.42	0.36
75%	0.91	0.39	0.91	0.84
max	1.85	2.18	1.48	1.91

In [29]:

```
logs1=LogisticRegression()
logs1.fit(x_train_sc,y_train)
logs1_pred=logs1.predict(x_test_sc)
cr1 = classification_report(y_test,logs1_pred)
cm1 = confusion_matrix(y_test,logs1_pred)
print("Report: ",cr1)
print("Matrix: ",cm1)
```

```
Report:                precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       0.83      0.62      0.71         8
           2       0.73      1.00      0.84         8

    accuracy                           0.87        30
   macro avg       0.85      0.85      0.84        30
weighted avg       0.88      0.87      0.86        30

Matrix:  [[13  1  0]
 [ 0  5  3]
 [ 0  0  8]]
```